REMARKS

Applicant has carefully considered the Office Action dated April 25, 2002 and the references cited therein. Applicant respectfully requests reexamination and reconsideration of the application.

Applicant proposes amendments to Figs. 1 and 5 to address the objections to the drawings as set forth in the Office Action. In addition, Applicant has amended the specification to eliminate nay reference numbers not shown in the drawings and to conform the description in the specification to Fig. 1,as originally filed. No new matter is believed added to the application by way of the proposed amendments to the figures as set forth herein.

Claims 1-30 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,263,379, Atkinson et al., hereafter Atkinson, in view of U.S. Patent No. 6,363,433, Nakajima. In setting for the rejection of claim 1, the Examiner has alleged that Atkinson discloses the claimed subject matter, however, the Examiner has admitted that Atkinson does not teach substituting a moniker object for the distributed object. Instead, the Examiner is relying on Nakajima to supply such teaching, alleging the Nakajima discloses a first stream object (MKParseDisplayName, line 46 column 5), which automatically substitutes the moniker object (76, Fig. 3) for the distributed object (72, Fig. 3) when the distributed object is streamed out (arrows going from 38 to 82, Fig. 3) from the memory (38, Fig. 3) to the local storage (82, Fig. 3). Th Examiner further asserts that it would have been obvious to apply the teachings of Nakajima to the system of Atkinson because the moniker object can be used in interfacing between the browser and the extension.

Applicant traverses the rejection of claims 1-30 under 35 U.S.C. §103(a) on the grounds that the Examiner has failed to create a *prima facie* case of obviousness. In accordance with MPEP §2143.03, to establish a *prima facie* case of obviousness 1) the prior art reference (or references when combined) must teach or suggest *all* of the claim limitations; 2) there must be some suggestion or motivation to modify a reference or combine references; and 3) there must be a reasonable expectation of success.

Applicant respectfully traverses the rejection of claims under 35 U.S.C. 103(a) on the grounds that the Examiner has failed to create a *prima facie* case of obviousness for failing to show how the prior art reference teach or suggest *all* of the claim limitations. Claim 1, as filed, recites an apparatus for providing naming and life cycle services for distributed objects including " a first stream object which automatically substitutes the moniker object for the distributed object when the distributed object is streamed out from the memory to the local storage" (Claim 1, lines 6-8). The Examiner has alleged that the Nakajima teaching of the MKParseDisplayName function (Nakajima, column 5, line 46) is analogous to the first stream object recited in claim 1. Applicant respectfully disagrees with the Examiner's analogy. In Nakajima, MKParseDisplayName is not an stream object, but an API function of the OLE standard, as set forth in Nakajima excerpts duplicated below:

> When a user selects that link, the browser 50 calls an OLE function known as MkParseDisplayName, passing the link's URL as a parameter. MkParseDisplayName parses the URL, creates a corresponding (aynchronous) URL moniker 54 therefrom, and returns a pointer to the moniker's IMoniker interface.
>
> (Nakajima, column 4, lines 39-43)

> [A] browser such as Internet Explorer 3.0 can create and plug-in an appropriate moniker using the OLE API function called MkParseDisplayName. More particularly, this function has the following form:
>
> HRESULT MkParseDisplayName(IBindCtx *pbc, LPCWSTR pszName, ULONG *pchEaten, IMoniker **ppmk)
>
> The MkParseDisplayName API function parses a text string (the parameter "pszName") into a moniker.
>
> (Nakajima, column 4, lines 39-43)

As such, the MKParseDisplayName function is not a streaming object, as recited in claim 1 and as described in the subject specification. The stream object of

claim 1 substitutes a moniker object for the distributed object when the distributed object is streamed out from the memory (Serial No. 09/244,291, Figs. 11-16, pp 20 *et seq*).

The Examiner has further alleged that the Nakajima teaching of extension 72 is the same as the distributed object recited in claim 1. Again, Applicant respectfully disagrees with the Examiner's analogy. In Nakajima, extension 72 is not a distributed object, but an Dynamic Link Library (DLL) according to the Microsoft Corporation's Internet Server Application Programming Interface (ISAPI) (Nakajima, column 1, line 65 to column 2, line 6). Nakajima further describes extension 72 in the excerpt below:

> In accordance with one aspect of the present invention, FIG. 3 shows a client computer system 20 having a client application 70 and a local ISAPI extension 72 present in the memory 24. As described above, the ISAPI extension 72 is a loaded DLL that executes according to whatever is coded therein. Indeed, the ISAPI extension 72 can be virtually identical to the exemplary ISAPI extension 62 described with respect to FIG. 2, i.e., it may service a user-query by first accessing the (local) database 38 to retrieve information therein related to the query, then ranking the information, and lastly formatting the result as an HTML document.
>
> (Nakajima, column 5, lines 14-24)

As such, the ISAPI extension disclosed by Nakajima is distinctly different than the distributed object recited in claim 1 and as described in the subject specification.

The Examiner has further alleged that extension control block 82, as described in Nakajima, is analogous to the local memory recited in claim1. Again, Applicant respectfully disagrees with the Examiner's analogy. In Nakajima, extension control block 82 is not a local memory as alleged by the Examiner, but is used for communication between the extension 72 and the moniker 76 as described in the Nakajima excerpts below:

> FIG. 4 shows the structure of an extension control block 82. The use of the extension control block 82 with respect to an HTTP server is well documented, and thus its use for communication between the extension 72 and the moniker 76 need not be described in detail herein. Note, however, that upon loading, the DLL should be initially called at the entry point of GetExtensionVersion to retrieve the version number of the

document on which the extension is based. This should be done to ensure proper initialization even though the retrieved information may be of no use to the local application 70. For every client request, the HttpExtensionProc entry point is called, (even though HTTP is not in use). In short, the ISAPI moniker 76 needs to appear to the extension 72 as a remote HTTP server, but can ignore any information returned from the extension that applies only to remote HTTP servers.

In any event, the extension receives commonly needed information such as the query string, path information, method name, and the translated path. To this end, the extension control block 82 includes fields such as cbSize enumerating the size of the block 82, dwversion, the version information of this document (the HIWORD contains the major version number), connID (IN), a unique number assigned by the HTTP server (or optionally, the moniker) and dwHttpStatusCode (OUT), the status when the current request is completed.

(Nakajima, column 6, lines 54 to column 7, line 12)

As such, the extension control block 82 disclosed by Nakajima is distinctly different than the local memory recited in claim 1 and described in the subject specification.

In light of the above, Applicants respectfully assert that the MKParseDisplayName OIE API function of Nakajima is not a first stream object, as recited in claim 1. The extension 72 of Nakajima is not a distributed object, as recited in claim 1. And, the extension control blocks 82 of Nakajima is not a local memory, as recited in claim 1.

In addition, claim 1 further recites a moniker object that "universally identifies an instance of the distributed object" (claim 1, lines 4-5). The section of Atkinson cited by the Examiner (Atkinson, column 10, lines 57-58) does not disclose a *universal* identifier as claimed and described in the subject specification.

Further, the Examiner will note that the motivation of the subject invention is not to facilitate an interface between browsers and the extensions, nor does such provide a motivation to combine the teachings of Atkinson and Nakajima. One of the object of the present invention is to provide distributed object systems with consistent naming and life cycle policies.

Accordingly, Applicant respectfully traverses the rejection of claims under 35 U.S.C. 103(a) on the grounds that the Examiner has failed to create a *prima facie* case of obviousness for : i) failing to show how the prior art reference teach or suggest *all* of the claim limitations, and ii) failing to show how some suggestion or motivation to combine the Atkinson and Nakajima references (MPEP §2143.03).

Claims 2-10 include all the limitations of claim 1 and are likewise believed allowable for at least the same reasons as claim 1.

Claim 11 includes limitation language similar to claim 1. Specifically, method claim 11 recites the step of " using a first stream object to automatically substitute the moniker object for the distributed object when the distributed object is streamed out from the memory to the local storage " (claim 11, lines 6-8). Accordingly, claim 11 is believed patentable over the teachings of Atkinson and Nakajima for at least the same reasons as claim 1. Claims 12-20 include all the limitations of claim 11 and are likewise believed allowable for at least the same reasons as claim 11.

Claim 21 includes limitation language similar to claims 1 and 11. Specifically, computer program product claim 21 recites " class code for instantiating a first stream object which automatically substitutes the moniker object for the distributed object when the distributed object is streamed out from the memory to the local storage" (claim 11, lines 6-8). Accordingly, claim 21 and its respective dependent claims are believed patentable over the teachings of Atkinson and Nakajima whether considered singularly or in combination with any other art of record, for at least the same reasons as claim 1.

Applicant believes the claims are in allowable condition. A notice of allowance for this application is solicited earnestly. If the Examiner has any further questions regarding this amendment, he/she is invited to call Applicant's attorney at the number listed below. The Examiner is hereby authorized to charge any fees or credit any balances under 37 CFR §1.17, and 1.16 to Deposit Account No. 02-3038.

Respectfully submitted,

Bruce D. Jobse Date: 7/23/02

Bruce D. Jobse, Esq. Reg. No. 33,518
KUDIRKA & JOBSE, LLP
Customer Number 021127
Tel: (617) 367-4600 Fax: (617) 367-4656

Version Marked to Show Changes

Paragraph beginning on page 7, line 11:

Figure 1 illustrates the a computer system comprising client computers 102, 104 and 106 intercoupled to each other and to a server 108 through a bus 105. Server 108 may be coupled to a database 110. Each of the computers may have a system architecture [for a computer system 100] such as an IBM PS/2®, on which the invention may be implemented. The exemplary computer system of Figure 1 is for descriptive purposes only. Although the description may refer to terms commonly used in describing particular computer systems, such as in IBM PS/2 computer, the description and concepts equally apply to other systems, including systems having architectures dissimilar to Figure 1.

Paragraph beginning on page 7, line 17:

Computer system 100 includes a central processing unit (CPU) [105] 118, which may be implemented with a conventional microprocessor, and a random access memory (RAM) [110] for temporary storage of information[,] and a read only memory (ROM) [115] for permanent storage of information, both collectively illustrated as memory 112. A memory controller [120] is provided for controlling [RMA 110] the RAM.

Paragraph beginning on page 7, line 22:

A bus [130] interconnects the components of each computer system [100]. A bus controller [125] is provided for controlling bus [130]. An interrupt controller [135] is used for receiving and processing various interrupt signals from the system components.

Paragraph beginning on page 7, line 25:

Mass storage may be provided by diskette [142], CD ROM [147], or hard drive [152]. Data and software may be exchanged with each computer system [100] via removable media such as diskette [142] and CD ROM [147]. The diskette [Diskette [42] is insertable into a diskette drive [141] which is, in turn, connected to the bus [30] by a controller [140]. Similarly, the CD ROM [147] is insertable into CD ROM drive [146] which is, in turn, connected to the bus [130] by a controller [145]. A hard [Hard] disk [152] is part of a fixed disk drive [151] which is connected to the bus [130] by controller [150].

Paragraph beginning on page 8, line 3:

User input to the computer [system 100] systems may be provided by a number of devices. For example, a keyboard 156 and mouse 157 are connected to the bus [130] by controller [155]. An audio transducer [196], which may act as both a microphone and a speaker, is connected to the bus [130] by an audio controller [197, as illustrated]. It will be obvious to those reasonably skilled in the art that other input devices, such as a pen and/or tabloid may be connected to the bus [130] and an appropriate controller and software, as required. A DMA controller [160] is provided for performing direct memory access to RAM [110]. A visual display is generated by video controller [165] which controls video display 170. Computer system 100 also includes a communications [adaptor 190] adapter which allows the system to be interconnected to a local area network (LAN) or a wide area network (WAN)[, schematically illustrated by bus 191 and network 195].

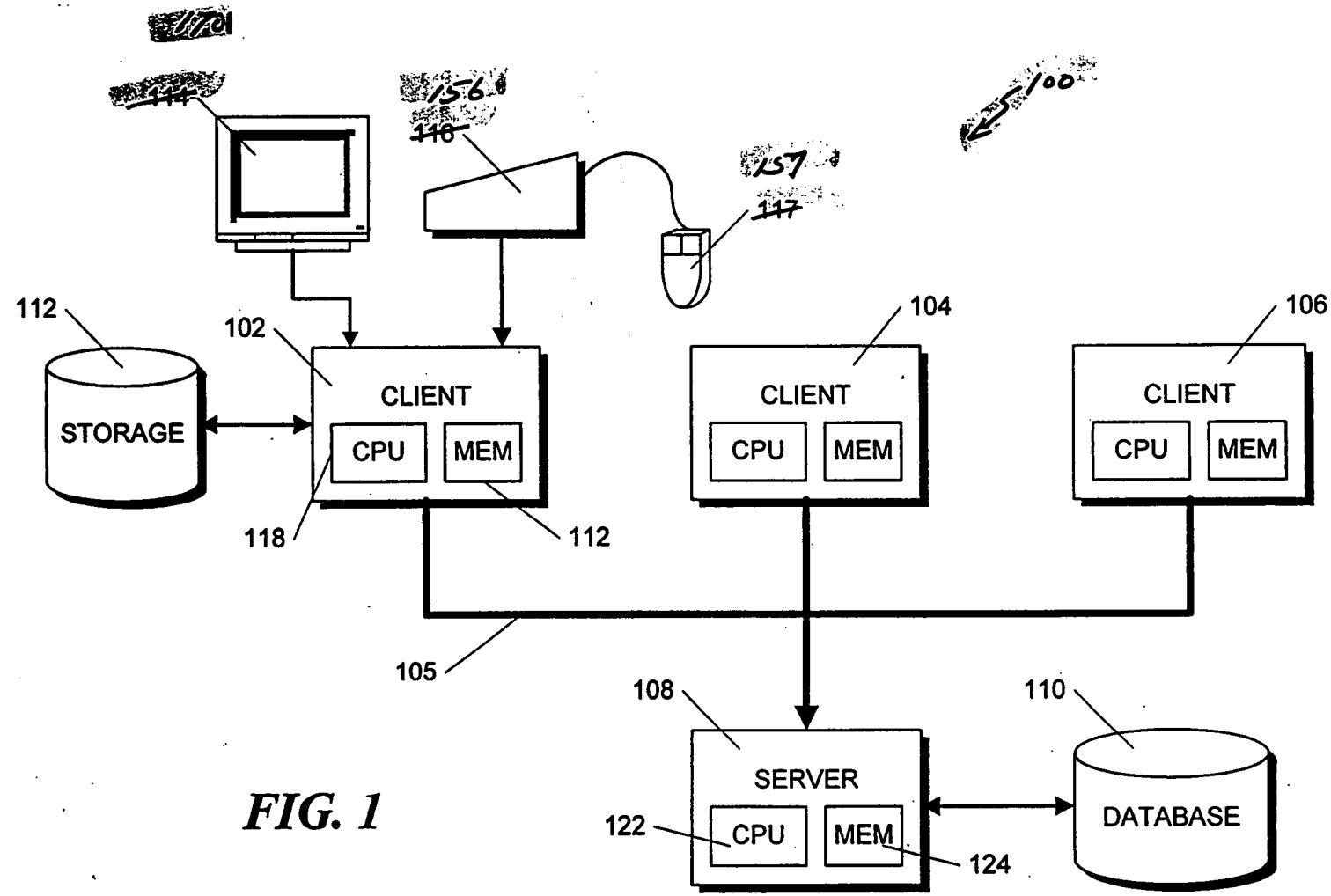
112
102
104
106
STORAGE
CLIENT
CPU
MEM
CLIENT
CPU
MEM
CLIENT
CPU
MEM
118
112
105
108
110
SERVER
CPU
MEM
DATABASE
122
124

FIG. 1

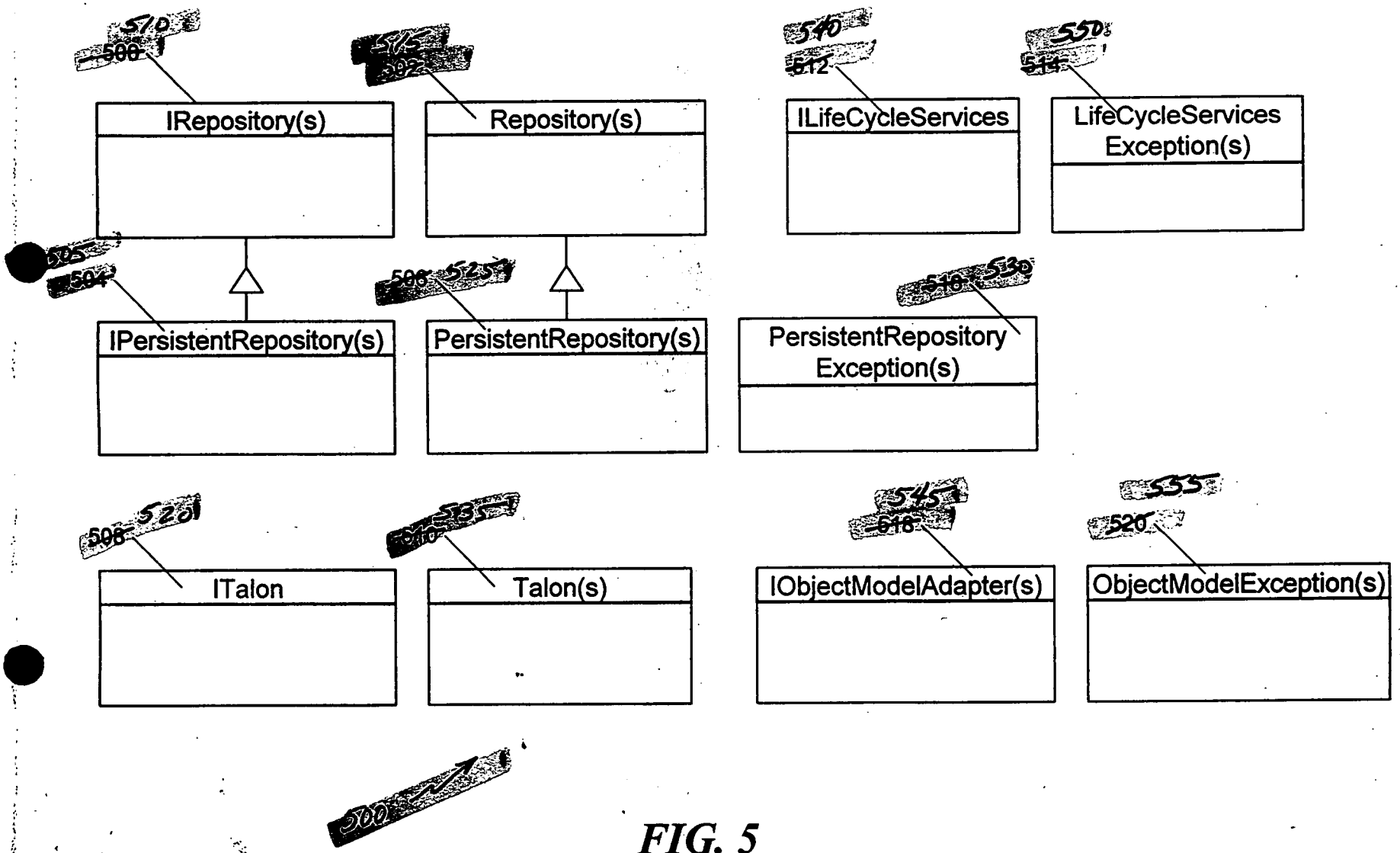
IRepository(s)
Repository(s)
ILifeCycleServices
LifeCycleServices Exception(s)
IPersistentRepository(s)
PersistentRepository(s)
PersistentRepository Exception(s)
ITalon
Talon(s)
IObjectModelAdapter(s)
ObjectModelException(s)

FIG. 5